Participating in Instructional Dialogues: Finding and Exploiting Relevant Prior Explanations

Participating in Instructional Dialogues: Finding and Exploiting Relevant Prior Explanations

JAMES A. ROSENBLUM
Department of Computer Science
University of Pittsburgh, Pittsburgh, Pa 15260, USA

JOHANNA D. MOORE
Department of Computer Science and Learning Research and Development Center
University of Pittsburgh, Pittsburgh, Pa 15260, USA
E-Mail: {jr, jmoore}@cs.pitt.edu

Abstract: In this paper we present our research on identifying and modeling the strategies that human tutors use for integrating previous explanations into current explanations. We have used this work to develop a computational model that has been partially implemented in an explanation facility for an existing tutoring system known as SHERLOCK.

We are implementing a system that uses case-based reasoning to identify previous situations and explanations that could potentially affect the explanation being constructed. We have identified heuristics for constructing explanations that exploit this information in ways similar to what we have observed in instructional dialogues produced by human tutors.

When human tutors engage in dialogue, they freely exploit all aspects of the mutually known context, including the previous discourse. Utterances that do not draw on previous discourse seem awkward, unnatural, or even incoherent. Previous discourse must be taken into account in order to relate new information effectively to recently conveyed material, and to avoid repeating old material that would distract the student from what is new. Thus, strategies for using the dialogue history in generating explanations are of great importance to research in natural language generation for tutorial applications.

The goal of our work is to produce a computational model of the effects of discourse context on explanations in instructional dialogues, and to implement this model in an intelligent tutoring system that maintains a dialogue history and uses it in planning its explanations. Based on a study of human-human instructional dialogues, we have developed a taxonomy that classifies the types of contextual effects that occur in our data according to the explanatory functions they serve (16). In this paper, we focus on one important category from our taxonomy: situations in which the tutor explicitly refers to a previous explanation in order to point out similarities (differences) between the material currently being explained and material presented in earlier explanation(s).

We are implementing a system that uses case-based reasoning to identify previous situations and explanations that could potentially affect the explanation being constructed. We have identified heuristics for constructing explanations that exploit this information in ways similar to what we have observed in instructional dialogues produced by human tutors. By building a computer system that has this capability as an optional facility that can be enabled or disabled, we will be able to systematically evaluate our hypothesis that this is a useful tutoring strategy.

Motivation and Problem Definition

In order to test our hypotheses about the effects of previous discourse on explanations, we are building an explanation component for an existing intelligent training system, SHERLOCK (11). SHERLOCK is an intelligent coached practice environment for training avionics technicians to troubleshoot complex electronic equipment. Using SHERLOCK, trainees solve problems with minimal tutor interaction and then review their troubleshooting behavior in a post-problem *reflective follow-up* session (RFU) where the tutor

TUTOR	<-> You tested pin 28 of the A1A3A15.	[1]
STUDENT	Why is testing pin 28 not a good step?	[2]
TUTOR	As explained before, the main data inputs to relay B24 are highly suspect at this time. You have tested the low input to B24 but not the high one. Since it is more likely that there may be a problem with the main data signals to relay B24 than with the secondary data signals to relays B25-B30, the main data signal should be tested first. Testing pin 28 is not a bad step, it is just more likely (on the average) that the main data signals are causing the problem.	[3]
	⋮ *One exchange and one action later.*	
TUTOR	<->You tested pin 36 of the A1A3A15.	[4]
STUDENT	Don't I need to test pin 36?	[5]
TUTOR	You might have to, but for the same reasons given when you tested pin 28, it is generally more efficient to test the main control data signals first, and then test the secondary control data signals if necessary.	[6]

Figure 1: *Human-Human Advisory Interaction Displaying Contextual Effects*

STUDENT	Why is testing pin 28 not a good step?
TUTOR	The main data inputs to relay B24 are highly suspect at this time. You have tested the low input to B24 but not the high one. Since it is more likely that there may be a problem with the main data signals to relay B24 than with the secondary data signals to relays B25-B30, the main data signal should be tested first. Testing pin 28 is not a bad step, it is just more likely (on the average) that the main data signals are causing the problem.
	⋮ *One exchange and one action later.*
STUDENT	Don't I need to test pin 36?
TUTOR	The main data inputs to relay B24 are highly suspect at this time. You have tested the low input to B24 but not the high one. Since it is more likely that there may be a problem with the main data signals to relay B24 than with the secondary data signals to relays B25-B30, the main data signal should be tested first. Testing pin 36 is not a bad step, it is just more likely that the main data signals are causing the problem.

Figure 2: *Hypothetical Advisory Interaction without Contextual Effects*

replays each student action and assesses it as "good" (<+>) or as "could be improved" (<->). After a step is replayed, the student can ask the tutor to justify its assessment. Psychological experimentation (14; 18) indicates that learning from task situations requires significant cognitive effort, and therefore some have argued that much of the instruction should actually take place in these post-problem RFU sessions in which students can review their own actions and compare them to expert behavior (10). Therefore, SHERLOCK does much of its tutoring during RFU where the most commonly asked question is for SHERLOCK to justify its assessment of a step (48% of RFU questions). Therefore, we have concentrated on handling this question type.

As an example of the way in which human tutors exploit previous discourse, consider the dialogue in Figure 1, taken from our data. Even though the student has made the same mistake twice, the second explanation looks quite different from the first. Yet the two explanations are related to one another in an important way. In the second explanation (turn 6) the tutor simply reminds the student that she has not determined the status of the main control data signals and that she should do so before testing the secondary control data signals. The tutor expects the student to be able to make use of the previous explanation (turn 3) once he has indicated that it is relevant to the current situation ("for the same reasons given..." serves this purpose). Accordingly, the tutor does not repeat the detailed explanation of why the main control data signals should be tested first. By generating the second explanation in such a way that it 'meshes' with the first, not only has the tutor corrected the testing mistake of the student,

but has forced the student to consider how the two situations are similar. In pointing out this similarity, he has given the student a better understanding of the domain[1]. We call an explanation that is later referred to (explicitly or implicitly) or is integrated into a subsequent explanation, the *anchor*.

Now consider what the response in turn 6 would look like if it were not generated in a manner that takes the explanation in turn 3 into account. (Figure 2). Here we see that the two explanations are nearly identical. If the student had trouble understanding the first response, repeating it a second time is unlikely to clear up her misunderstanding. Moreover, in this dialogue, it is much more difficult for the student to recognize that the two situations are similar and are both instances of the same general type of error.

We have observed that tutors regularly produce explanations that make use of the previous discourse. They explain general principles and then instantiate them with instances from the student's own behavior in the current problem-solving situation. They refer to previous explanations to point out similarities and differences between situations, to avoid repeating material that would distract the student from what is new, or to try an alternative explanation if a previous explanation is not satisfactory.

Producing a system that displays such behavior requires tackling two problems: (1) deciding what previous explanation (or part thereof) to use as an anchor, and (2) determining how the selected anchor should affect the construction of the current explanation. The first problem involves deciding, *in an efficient way*, whether there exist suitable candidates to act as anchor, and if so, which amongst them would be best to use. The second problem is to determine how to modify the current explanation in a pedagogically beneficial way, based on the previous explanation. In the following sections we introduce our domain and describe our data gathering, show how we have solved the first of these problems, and present our preliminary solutions to the second. Finally, we put this work in the context of the larger research effort of which it is part.

Overview of Sherlock

As mentioned above, we are working in the context of SHERLOCK, an existing intelligent system that trains students to troubleshoot a complex electronic device. SHERLOCK is a realistic computer simulation of the actual job environment. Trainees acquire and practice skills in a context similar to the actual setting in which they will be used. During a problem-solving session, SHERLOCK's intelligence is used mainly to provide hints in response to student requests rather that to intervene actively. Help is provided only on request or when the trainee's performance requires SHERLOCK's intervention (e.g., when he or she attempts an unsafe action). If trainees do not know what to do next or if they do not know how to perform a task, they may ask for help.

After students have solved a problem, they can use RFU to reflect on their performance by reviewing their work with the assistance of the computer-based coach. The purpose of RFU is to help students internalize the curriculum issues that SHERLOCK teaches. During RFU, students can replay their actions step-by-step. As each step is replayed, a color-coded diagram shows what can be deduced about the system's circuitry from the troubleshooting steps performed thus far. In addition, SHERLOCK marks the student's step as "good" or "could be improved." In the current system, there is a menu of questions that students can ask along the way, including questions about the tutor's assessment of their actions, what an expert would do at a particular point, how the system knows that a component is good, etc.

Currently SHERLOCK's RFU facility responds to student's questions by filling in templates based on the question type and particulars of the student's action and the problem situation. This type of explanation facility is designed by identifying a set of situations as being ones where text should be produced, hand-crafting templates for those situations, and indexing the templates so that the correct one can be printed at the appropriate time. This approach exhibits a number of shortcomings. First, in order to keep the task of hand-crafting templates manageable, one must severely restrict the number of features that describe a situation. Second, since text is not produced directly by procedures that examine system-internal knowledge structures, this approach cannot guarantee that explanations match program code as the system is modified over time. Finally and most limiting, since a system using templates has no model of what it is saying, it cannot reason about the explanations it has given. Therefore, it cannot generate explanations that take into account prior explanations.

Although space constraints prevent a complete discussion of the way SHERLOCK reasons about the domain and student actions, the following aspect is extremely important to our work, and so we briefly

[1] Notice that the first explanation (turn 3) makes use of a previous explanation (signaled by, "as explained before"), not shown due to space constraints.

describe it here. SHERLOCK evaluates each student action by determining which *facets* apply to that action. The facets come from a cognitive task analysis aimed at identifying the factors that expert avionics tutors use in assessing student's troubleshooting actions (15). To evaluate an action, SHERLOCK finds each facet that applies to it and determines whether that facet should be considered good (g), bad (b), or neutral (n) in the current problem-solving context. Some facets characterize aspects of actions that are *always* considered good or bad. For example, the facet "Making a measurement that is off the active circuit path" is always considered a b-facet. The representation of a student action includes the list of facets characterizing the action and an assessment (g, b, or n) for each of those facets.

Analysis of Human-Human Reflective Dialogues

In order to identify the strategies that tutors use in generating instructional explanations, we collected samples of human-human interactions in the following way. Each student was asked to solve a problem using SHERLOCK. Afterwards, students used the RFU facility to review their problem-solving. During our experiments, students were not allowed to question the computer about their performance. Instead, they addressed any questions they had to a human expert tutor. For the experiments, the SHERLOCK image was displayed onto two monitors, so that the tutor and student could be physically arranged so as to inhibit communication by speaking or gesturing. The tutor sat approximately 6 feet from the student, and observed all of the student's actions on the second screen. Whenever the student wished to communicate with the tutor, he or she wrote the question on a pad of paper, and passed it to the tutor. The tutor then wrote an answer on the pad, and passed it back to the student. Students were instructed to ask as many questions as they liked. The only constraint was that the student and tutor did not communicate except by writing on the pad.

To date, we have collected data from 24 student-tutor interactions with 14 different students and 3 different tutors. In total we have examined approximately 2024 sentences containing 518 question/answer pairs. Of these, 990 sentences and 232 question/answer pairs took place in RFU. This study indicated to us that tutors frequently integrate previous explanations into their answers, and we hypothesize that this capability is essential for effective explanation. By building a system that has this ability as an optional facility that can be enabled or disabled, we will be able to systematically evaluate this hypothesis.

Finding a Relevant Prior Explanation

To produce explanations that take into account prior discourse, the system must be able to identify situations in which one of its prior utterances could affect the current response. In order to do this, a system must represent the text it produces in such a way that it can determine what effect each part of the explanation was intended to have on the student. In addition, the system must maintain a record of the explanations it has produced and must organize this record in such a way that it can be efficiently examined when looking for appropriate prior explanations.

The Text Planner

For this project, we are extending the text planner built by Moore and Paris (1989), which works in the following way. When the user provides input to the system, the query analyzer interprets the question and forms a *communicative goal*, e.g., "achieve the state where the hearer believes that an action could be improved," or "achieve the state where the hearer understands how the status of a component has been determined." The text planner uses a linear planning mechanism to synthesize responses to achieve these goals. The system's knowledge about explanation is contained in a library of plan operators that map communicative goals to linguistic resources (speech acts and rhetorical strategies) for achieving them. When a communicative goal is posted, the planning mechanism uses its operators to construct a *text plan* for an explanation. In general, there may be many plan operators available to achieve a given goal, and the planner has a set of *selection heuristics* for choosing among them. Planning is complete when all goals have been refined into speech acts, such as INFORM and RECOMMEND.

The system then presents the explanation to the user, retaining the plan that produced it in a *dialogue history*. The dialogue history is a record of the conversation that has occurred thus far and includes the user's utterances and the text plans that led to the system's responses. In this system, a text plan represents the effect that each part of the text is intended to have on the hearer's mental state, the linguistic strategies that were used to achieve these effects, and how the complete text achieves the overall communicative goal.

Knowledge Sources for Finding Relevant Prior Explanations

The most straightforward way to find relevant prior explanations would be to search the system's dialogue history, beginning at the most recently presented explanation and working back in time through the dialogue, looking for explanations that have certain features. For example, when explaining why a step was assessed as "could be improved," the system could look for previous explanations that justified a "could be improved" assessment, and in which the two actions being assessed were similar (i.e., had the same facets).

There are two problems with this approach. First, explanation plans are large, complex structures, and they accumulate rapidly as the dialogue progresses. Therefore, exhaustively searching the dialogue history is computationally expensive. Second, there are some situations in which there is no text plan to be found in the dialogue history. For example, the student may have previously performed an action that displayed some characteristic(s) that the tutor decided not to mention at the time and which would now be appropriate to talk about in conjunction with the student's current action.

Therefore, we require an indexing strategy that will allow the system to find possibly relevant explanations in the dialogue history in an efficient manner. In addition, the system must have the capability to find relevant prior actions even if the aspects of those actions that are relevant to the current situation were never mentioned. To satisfy these requirements, we use case-based reasoning to provide a framework in which previous student actions can be efficiently examined to determine which, if any, are relevant when producing an explanation. This framework, and its associated indices into the dialogue history, allow both what *was* and *was not* said to be considered when the system plans an explanation.

A Case-Based Algorithm

Case-Based Reasoning (CBR) generalizes from cases to support indexing and relevance assessment, and can be used to evaluate a case by comparing it to past cases (4). This describes our task if we treat each student action as a "case". We were influenced by the work of Ashley (1990) who built a system, HYPO, that analyzes a legal case by comparing it to the cases in its case library based on similarity factors. Each factor is considered to support either the defendant or the plaintiff. The representation of a case includes a set of applicable factors plus an outcome for defendant or plaintiff. HYPO uses this information to construct a *claim lattice* which represents a partial ordering of a set of past cases in terms of their similarity to a case to be decided. HYPO uses the claim lattice to find the most relevant case to cite in a legal argument. Aleven and Ashley (1992) have extended this work to a tutorial context. In our system, student actions are assessed an outcome of either "good" or "could be improved," and SHERLOCK evaluates each action with respect to a set of facets with an associated judgement about whether the facet supports a "good" or "could be improved" outcome. Thus, we can use CBR techniques to identify similar actions as described below.

Our algorithm builds a data structure called a *similarity DAG* (Directed Acyclic Graph) which indicates the previous student actions that are similar to a given action. By similar, we mean similar with respect to a certain class of facets (some combination of g, b, or n). For example, when answering a question about why the current action was assessed as "could be improved," the similarity DAG is built so that it indicates which previous actions were similar to the current action with respect to the b-facets. The root of the DAG represents the current action and the facets of interest (b-facets in our example) that apply to it. Each node in the DAG represents a set of student actions that share the same set of interesting facets. The more facets that a node has in common with the current action (the root node), the closer it will be to the root node. Proximity in the DAG corresponds to similarity in facet sets. Basically, the similarity DAG is a partial ordering of the student's actions based on their facet lists.

Figure 3 shows the similarity DAG (consisting of two nodes) that is constructed when the system considers how to answer the question, "Don't I need to test pin 36?" (turn 5 of the sample dialogue shown in Figure 1). The facets relevant to the action in question are F100 and F101. The structure indicates that two previous situations are similar to the current situation – action 9 and to a lesser degree action 8. Pointers index the dialogue history's record of what was said at those times. At this point, the system has identified candidate situations which may be used in planning the current explanation. It can now consider these retrieved situations more closely to determine any other facets that they may possess, and can examine the related explanations in the dialogue history to determine what was said about each of the two previous situations. The fact that there are no other nodes in the DAG indicates that there are no other suitable prior situations. If actions 8 and 9 had not occurred, the DAG would consist of one

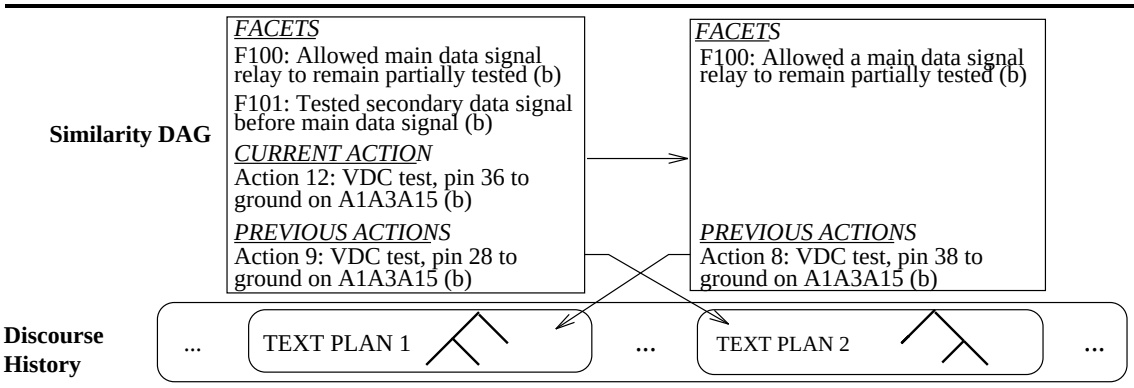

Figure 3: *Data structures when considering how to answer turn 5, Figure 1*

empty root, indicating that no prior action was similar to the current action in terms of its assessment.

Initial results using this algorithm are quite promising. In an analysis of 8 student-tutor protocols involving 154 actions and 22 opportunities for integrating a previous explanation into an answer, the algorithm correctly identified the same previous situations that were used by the human tutor in the actual interaction. In all but 3 cases, when the human tutor did not make a reference to a previous explanation, our algorithm reported no similar prior situation. In the 3 situations where our algorithm identified a similarity not exploited by the tutor, our expert agreed that the associated prior explanations would have been useful to incorporate into his subsequent explanations.

Finally, this technique will be useful in answering students' direct questions about the similarities of situations, e.g., "Why is testing 30 good? Isn't it like 36 and 28?" By constructing and consulting a similarity DAG, the system is able to plan responses such as: "Yes, but now you know the main control data signals on pins 33 and 22 are good, so you need to test the secondary data signals."

Exploiting the Previous Explanation

Having found a suitable situation and associated text plan, the system must decide how to use this information when generating the current explanation. This task is, by far, the harder of the two problems and is the topic of Rosenblum's thesis. However, as a starting point, we have developed a simple algorithm that is capable of reproducing some of the effects we have observed in human tutors' explanations.

Recall that when answering the question regarding why testing pin 36 was judged as "could be improved" (turn 5 of Figure 1), the CBR-algorithm indicated that two previous student actions were relevant - action 9 (testing pin 28) and to a lesser degree action 8 (testing pin 38). Accordingly, the student action of testing pin 28 is judged more relevant to the current situation. The representation of this action includes a pointer to the text plan in the dialogue history representing the explanation given at that time (Figure 4). This text plan will be consulted as the answer to the student's question about testing pin 36 (turn 5 of Figure 1) is formulated.

The system begins by generating the top-level goal of having the hearer know why the assessment of testing pin 36 is "could be improved." This matches the top-level goal posted in the previous explanation's text plan. The system then refines this top-level goal into a subgoal to make the hearer believe that testing the inputs to B24 is a better step. When consulting the prior text plan, the system discovers that it had refined the previous explanations' top-level goal in a similar way. In fact, simple pattern matching reveals that the only difference is that previously the student had tested pin 28 and this time the student has tested pin 36. An examination of SHERLOCK's knowledge representation reveals that both pin 28 and pin 36 are secondary data signals to relays on the same card, so this difference is not important in terms of the student's mistake. This strong match between the system's developing text plan and the beginning of the prior text plan, causes the system to generate the goals of anaphorically referencing and of summarizing the previous explanation resulting in, "You might have to, but for the same reasons given when you tested pin 28, it is generally more efficient to test the main control data signals first, and then test the secondary control data signals if necessary.

To date we have devised heuristics that handle a small number of examples such as the one above, and we are currently in the process of implementing them.

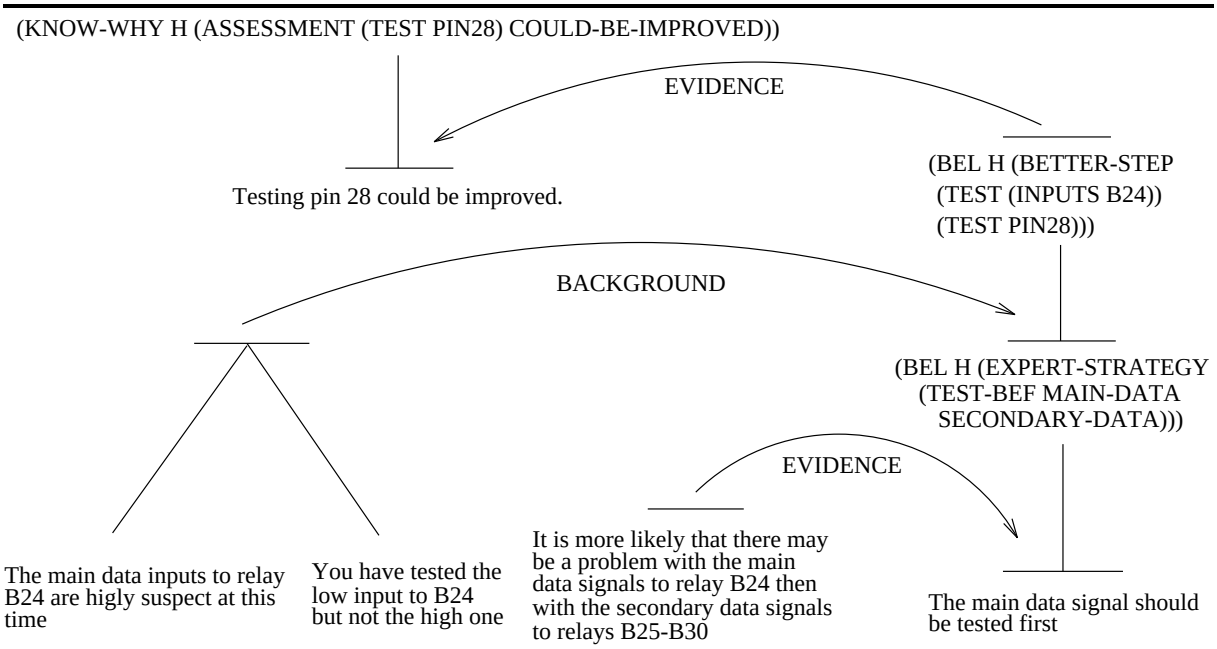

Figure 4: *RST discourse tree for turn 3, Figure 1.*

Related Work

Computational linguists have investigated how the context provided by the previous discourse should affect the generation of referring expressions, including pronominalization decisions (8; 7). Others have studied how a more extensive discourse history could affect other aspects of the response. Swartout's (1983) XPLAIN system can suggest simple analogies with previous explanations and omit portions of a causal chain that have been presented in an earlier explanation. However, this is the only type of contextual effect implemented in XPLAIN, and it was done using an ad hoc technique to provide this one effect. We are attempting to provide a more general approach.

McKeown (1985) carried out a preliminary analysis of how previous discourse might affect a system's response to users' requests to describe an object or compare two objects. She found that by simply maintaining a list of the questions that had been asked, it was possible to avoid certain types of repetition. She further found that if the system were to keep track of the exact information that was provided previously, it could create a text that contrasts or parallels an earlier one. While McKeown's analysis was fairly detailed, no dialogue history was maintained in the implementation, and none of the suggestions for how responses could be altered, if such a history existed, were actually implemented or tested. We are devising explanation strategies that allow a system to make use of the information stored in a dialogue history, and are implementing these strategies.

Finally, some of our heuristics for utilizing prior explanations resemble work done in *plan adaptation* (3). For example, the heuristic we described above can be viewed as modifying a retrieved plan to fit a new situation. In addition, systems using plan adaptation often use CBR techniques to index a library of plans that can be adapted. Plan adaptation is concerned with indexing plans so that they can be retrieved and reused, perhaps with some modifications, in later situations. Our emphasis is not on finding prior explanations so that they may be reused, but rather so that they can be exploited as one of many knowledge sources affecting explanation generation.

In addition to the use of heuristics, we are exploring the use of plan operators whose constraints check for certain patterns in the user model and dialogue history. Carenini and Moore (1993) have shown that this approach can be used to generate explanations that are sensitive to the context created by prior explanations. However, as they point out, this approach may lead to a proliferation of plan operators. We are currently exploring the use of context-sensitive plan operators, plan selection heuristics, and "plan critics" in an integrated approach.

Conclusion and Future Work

We have argued that human tutors give instruction in a way that is sensitive to what has been previously said and done. We have suggested that a dialogue history, alone, is insufficient to allow a system to take into account important contextual issues. Issues such as what was not said in a particular situation or what event, if any, would be useful to refer to in an answer. A case-based reasoning approach was put forth as a method of identifying relevant prior explanations when addressing questions regarding the system's assessment of a student's action. Plan adaptation heuristics were then shown to be useful in performing the actual integration of the retrieved situation (and associated text) into the current explanation.

Many of the issues presented in this paper deserve, and will receive, future work. In addition to further evaluation of our algorithm for finding similar situations, we will immediately concentrate on extending our heuristics for generating explanations under the influence of relevant prior explanations. We will also examine the interactions between explanations responding to different question types. This work is part of a larger project whose goals were alluded to in the beginning of the paper. We are attempting to categorize the ways in which humans exploit previous discourse in instructional dialogue and build a computational model that can be used in intelligent training systems.

References

Aleven, V. & Ashley, K.D. (1992). Automated generation of examples for a tutorial in case-based argumentation. In C. Frasson, G. Gauthier, G. McCalla editors, *Proceedings of the* 2^{nd} *Int'l Conference on Intelligent Tutoring Systems*, 575-584. Springer-Verlag, Berlin.

Alterman, R. (1992). Adaptive planning. In Stuart Shapiro, editor, *The Encyclopedia of Artificial Intelligence*, pages 5–15. Wiley, New York.

Ashley, K.D. (1990). *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* MIT Press, Cambridge, Chapter 8.

Ashley, K.D. (1992). Case-based reasoning and its implications for legal expert systems. *Artificial Intelligence and Law*, 2(1).

Carenini, G. & Moore, J.D. (1993). Generating explanations in context. In *Proceedings of the Int'l Workshop on Intelligent User Interfaces*, Orlando, Florida, January.

Dale, R. (1989). Cooking up referring expressions. In *Proceedings of the* 27^{th} *Annual Meeting of the Association for Computational Linguistics*, pages 68–75, Vancouver, B.C., Canada, June 26-29.

Granville, R. (1984). Controlling lexical substitution in computer text generation. In *Proceedings of the* 10^{th} *Int'l Conference on Computational Linguistics*, pages 381–384, Stanford University, July.

Lesgold, A. (1990). Assessment of intelligent training technology. In *Proceedings of the Office of Naval Research Conference on Technology Assessment*, Los Angeles, California, September.

Lesgold, A., Lajoie, S., Bunzo, M. & Eggan, G. (1992). Sherlock: A coached practice environment for an electronics troubleshooting job. In *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

McKeown, K.R. (1985). *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press, Cambridge, England.

Moore, J.D. & Paris, C.L. (1989). Planning text for advisory dialogues. In *Proceedings of the* 27^{th} *Annual Meeting of the Association for Computational Linguistics*, pages 203–211, Vancouver, B.C., Canada.

Owen, E. & Sweller, J. (1985). What do students learn while solving mathematics problems? *Journal of Educational Psychology*, 77:272–284.

Pokorny, R. & Gott, S. (1990). The evaluation of a real-world instructional system: Using technical experts as raters. Technical report, Armstrong Laboratories, Brooks AFB.

Rosenblum, J.A. & Moore, J.D. (forthcoming). A field guide to contextual effects in instructional dialogues. Technical report, University of Pittsburgh, Computer Science Department.

Swartout, W. R. (1983). XPLAIN: A system for creating and explaining expert consulting systems, *Artificial Intelligence*,Vancouver, B.C., Canada, 21(3), 285–325.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12:257–285.